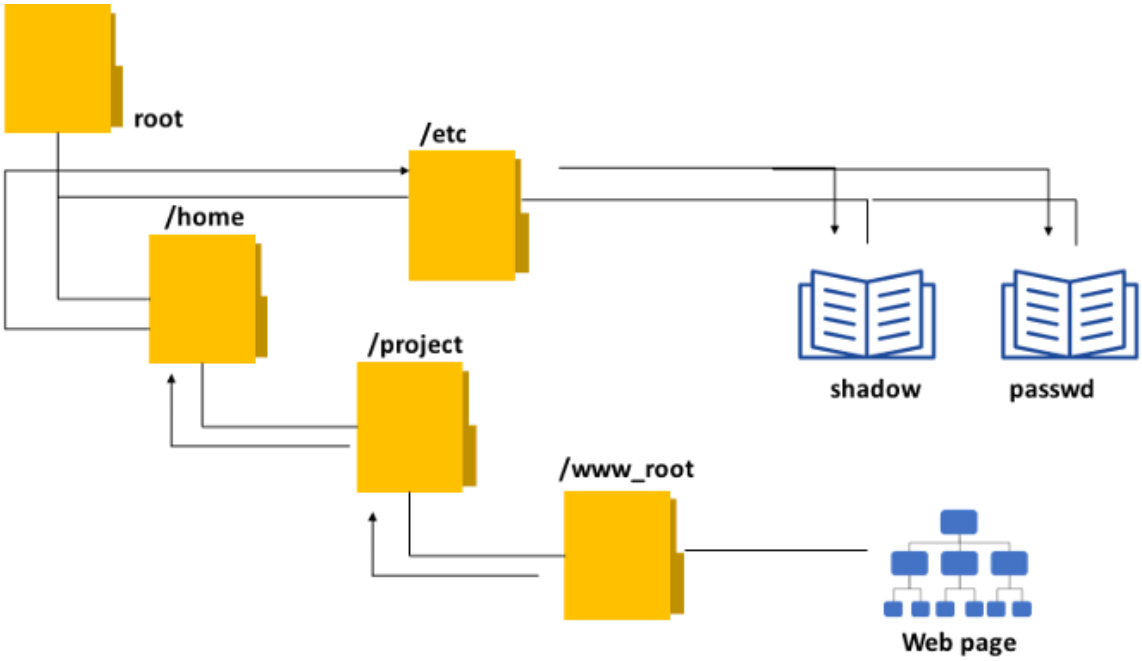
File Path Traversal/File Inclusion and File Upload Vulnerabilities



What is File Path Traversal?

Directory traversal (*also known as file path traversal*) is a web security vulnerability that allows an attacker to read arbitrary files on the server that is running an application. This might include application code and data, credentials for back-end systems, and sensitive operating system files. In some cases, an attacker might be able to write to arbitrary files on the server, allowing them to modify application data or behaviour, and ultimately take full control of the server.

Example: Let's say you have a website running on http://www.example.com. Let's also suppose that the web server you are using makes it super easy to add pages to your site; all you have to do is add them to the web root folder, /var/www, on the server's filesystem and the rest is taken care of. If you add the

file /var/www/products/table.html, then that page can be accessed by anyone if they visit http://example.com/products/table.html. This web server, unfortunately, is super old and vulnerable to path traversal. This allows an attacker to use special character sequences, like ../, which in Unix directories points to its parent directory, to traverse up the directory chain and access files outside of /var/www, like this.

When receiving this request, the web server appends the relative path specified by the user, ../../configuration.yml, to the directory that holds the web pages, /var/www/, to obtain the full path /var/www/../../configuration.yml. In Unix-like systems, each ../ cancels out the directory immediately to the left of it, so if we reduce the path to its simplified form, the final path becomes /private/configuration.yml.

And now, the hacker has just obtained sensitive information, maybe even your database credentials, and can use this information to steal your users' information or cause further damage.

Impact

An attacker can leverage a directory/file traversal vulnerability in the system to step out of the root directory, allowing them to access other parts of the file system to view restricted files and gather more information required to further compromise the system.

What are File Inclusion Vulnerabilities?

File Inclusion vulnerabilities often affect web applications that rely on a scripting run time, and occur when a web application allows users to submit input into files or upload files to the server. They are often found in poorly-written applications.

File Inclusion vulnerabilities allow an attacker to read and sometimes execute files on the victim server or, as is the case with Remote File Inclusion, to execute code hosted on the attacker's machine.

An attacker may use remote code execution to create a web shell on the server, and use that web shell for website defacement.

Types of file inclusion vulnerabilities

File inclusion vulnerabilities come in two types, depending on the origin of the included file:

- Local File Inclusion (LFI)
- Remote File Inclusion (RFI)

Local File Inclusion (LFI)

A Local File Inclusion attack is used to trick the application into exposing or running files on the server. They allow attackers to execute arbitrary commands or, if the server is misconfigured and running with high privileges, to gain access to sensitive data.

Remote File Inclusion (RFI)

An attacker who uses Remote File Inclusion targets web applications that dynamically reference external scripts. The goal of the attacker is to exploit the referencing function in the target application and to upload malware from a remote URL, located on a different domain.

Impact

- An attacker may use remote code execution to create a web shell on the server, and use that web shell for website defacement.
- Code execution on the client-side such as JavaScript can lead to other attacks such as cross-site scripting (XSS)
- Sensitive Information Disclosure
- Denial of Service (DoS)
- Data Manipulation Attacks

File upload vulnerability

File upload vulnerabilities are when a web server allows users to upload files to its filesystem without sufficiently validating things like their name, type, contents, or size. Failing to properly enforce restrictions on these could mean that even a basic image upload function can be used to upload arbitrary and potentially dangerous files instead. This could even include server-side script files that enable remote code execution.

In some cases, the act of uploading the file is in itself enough to cause damage. Other attacks may involve a follow-up HTTP request for the file, typically to trigger its execution by the server.

What is the impact of file upload vulnerabilities?

The impact of file upload vulnerabilities generally depends on two key factors:

- Which aspect of the file the website fails to validate properly, whether that be its size, type, contents, and so on.
- What restrictions are imposed on the file once it has been successfully uploaded.

In the worst-case scenario, the file's type isn't validated properly, and the server configuration allows certain types of files (such as .php and .jsp) to be executed as code. In this case, an attacker could potentially upload a server-side code file that functions as a web shell, effectively granting them full control over the server.

If the filename isn't validated properly, this could allow an attacker to overwrite critical files simply by uploading a file with the same name. If the server is also vulnerable to directory traversal, this could mean attackers are even able to upload files to unanticipated locations.

Failing to make sure that the size of the file falls within expected thresholds could also enable a form of denial-of-service (DoS) attack, whereby the attacker fills the available disk space.

References

- **Portswigger.com**
- **synopsys.com**
- **brightsec.com**
- **brightsec.com**